ESEMPIO SERVIZIO CLIENT-SERVER

Ecco un esempio di codice di un semplice applicativo lato client che effettua chiamate ad un server che ha un web service REST:

```
<!DOCTYPE html>
<html>
<head>
  <title>Applicativo Client REST</title>
  <script>
    function effettuaChiamata() {
      // Ottenere i valori dei parametri dall'utente
      var nome = document.getElementById("nome").value;
      var email = document.getElementById("email").value;

      // Validare i parametri con espressioni regolari
      var regexEmail = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!regexEmail.test(email)) {
        alert("Inserire un indirizzo email valido");
        return;
      }

      // Effettuare la chiamata REST al server utilizzando il metodo POST
      var url = "https://mioserver.com/api/utenti";
      var params = {
        nome: nome,
        email: email
      };
      fetch(url, {
        method: "POST",
        body: JSON.stringify(params),
        headers: {
          "Content-Type": "application/json"
        }
      })
      .then(function(response) {
        if (response.ok) {
          return response.json();
        } else {
          throw new Error("Errore nella chiamata al server");
        }
```

```
      })
      .then(function(data) {
        // Elaborare la risposta del server
        var idUtente = data.id;
        var messaggio = "L'utente " + nome + " è stato creato con successo con l'ID " + id
Utente;
        alert(messaggio);
      })
      .catch(function(error) {
        // Gestire eventuali errori
        alert(error.message);
      });
    }
  </script>
</head>
<body>
  <h1>Creazione di un nuovo utente</h1>
  <label>Nome:</label>
  <input type="text" id="nome"><br><br>
  <label>Email:</label>
  <input type="text" id="email"><br><br>
  <button onclick="effettuaChiamata()">Crea utente</button>
</body>
</html>
```

In questo esempio, l'applicativo client HTML consente all'utente di inserire il nome e l'indirizzo email di un nuovo utente e di inviarli al server utilizzando una chiamata REST con il metodo POST. Prima di effettuare la chiamata, l'applicativo controlla che l'indirizzo email inserito sia valido utilizzando un'espressione regolare.

Il parametro JSON contenente il nome e l'indirizzo email dell'utente viene inviato al server nel corpo della richiesta POST con l'header Content-Type impostato su application/json.

Il server REST riceve la richiesta POST dal client e crea un nuovo record nella tabella Utenti del database, utilizzando i parametri ricevuti dal client. Il server restituisce poi una risposta al client, contenente l'ID del nuovo utente creato. La risposta viene inviata in formato JSON.

Il client riceve la risposta dal server e la elabora, estraendo l'ID dell'utente creato e mostrando un messaggio di conferma all'utente. Se si verifica un errore durante la chiamata REST, il client gestisce l'errore mostrando un messaggio

```
using System;
using System.Net;
using System.Web.Script.Serialization;
```

```
public class Utente {
    public int Id { get; set; }
    public string Nome { get; set; }
    public string Email { get; set; }
}

public class Server {
    private static int idUtente = 1; // variabile per tenere traccia dell'ID dell'utente

    public static void Main(string[] args) {
        // Configurare il server REST per ascoltare le richieste POST sulla URL /api/utent
i
        HttpListener listener = new HttpListener();
        listener.Prefixes.Add("http://mioserver.com:8080/api/utenti/");
        listener.Start();
        Console.WriteLine("Server in ascolto su http://mioserver.com:8080/api/utenti/");

        // Gestire le richieste POST
        while (true) {
            HttpListenerContext context = listener.GetContext();
            if (context.Request.HttpMethod == "POST") {
                // Leggere i parametri JSON dalla richiesta POST
                var serializer = new JavaScriptSerializer();
                var body = context.Request.InputStream;
                var encoding = context.Request.ContentEncoding;
                var reader = new System.IO.StreamReader(body, encoding);
                var requestBody = reader.ReadToEnd();
                var parametri = serializer.Deserialize<Utente>(requestBody);

                // Creare un nuovo utente con i parametri ricevuti dal client
                Utente nuovoUtente = new Utente {
                    Id = idUtente,
                    Nome = parametri.Nome,
                    Email = parametri.Email
                };
                idUtente++;

                // Inserire il nuovo utente nel database
                InserisciUtenteNelDatabase(nuovoUtente);

                // Restituire l'ID del nuovo utente creato al client
                var response = context.Response;
                response.ContentType = "application/json";
                response.StatusCode = (int) HttpStatusCode.OK;
                var responseBody = new JavaScriptSerializer().Serialize(new { id = nuovoUt
ente.Id });
                var buffer = System.Text.Encoding.UTF8.GetBytes(responseBody);
                response.ContentLength64 = buffer.Length;
                var output = response.OutputStream;
                output.Write(buffer, 0, buffer.Length);
                output.Close();
            }
        }
```

```
    }

    private static void InserisciUtenteNelDatabase(Utente utente) {
        // Connessione al database
        string connectionString = "server=miodatabase.com;database=miodatabase;uid=user;pw
d=password;";
        using (MySqlConnection conn = new MySqlConnection(connectionString)) {
            conn.Open();

            // Inserimento del nuovo utente nella tabella Utenti
            MySqlCommand command = new MySqlCommand("INSERT INTO Utenti (Id, Nome, Email)
 VALUES (@Id, @Nome, @Email)", conn);
            command.Parameters.AddWithValue("@Id", utente.Id);
            command.Parameters.AddWithValue("@Nome", utente.Nome);
            command.Parameters.AddWithValue("@Email", utente.Email);
            command.ExecuteNonQuery();
        }
    }
}
```

In questo esempio, il server REST è configurato per ascoltare le richieste POST sulla URL **http://mioserver.com:8080/api/utenti/**.

Quindi, il server restituisce l'ID del nuovo utente creato al client in formato JSON. Il client può quindi utilizzare questo ID per effettuare altre richieste al server per recuperare o aggiornare le informazioni sull'utente.

Nel codice del server, la gestione della richiesta POST avviene nel ciclo while, che ascolta continuamente le richieste HTTP in arrivo tramite la classe HttpListener. Quando viene rilevata una richiesta POST, il server legge i parametri JSON dal corpo della richiesta, li deserializza in un oggetto Utente e lo inserisce nel database chiamando il metodo InserisciUtenteNelDatabase. Infine, il server restituisce un oggetto JSON contenente l'ID del nuovo utente creato al client.

Il codice utilizza il driver MySQL Connector/NET per effettuare la connessione al database MySQL e interagire con esso. Per la gestione delle richieste HTTP, viene utilizzata la classe HttpListener di .NET.